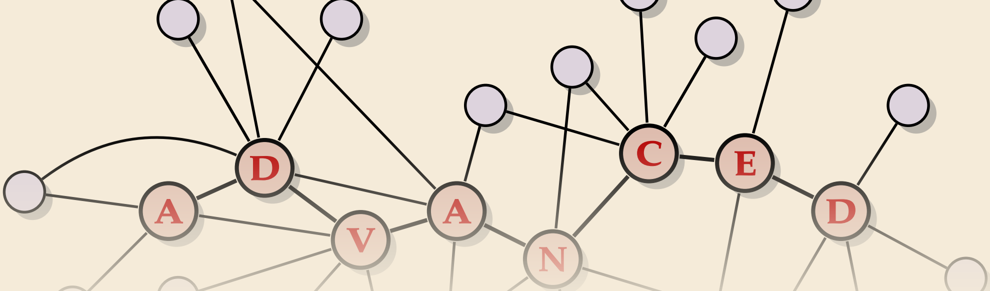

8 Randomized Complexity

17 June 2025

Prof. Dr. Sebastian Wild

CS627 (Summer 2025)
Philipps-Universität Marburg
version `2025-06-12 00:04 H`

Outline

8 Randomized Complexity

8.1 Randomized Complexity Classes

Does randomization extend the range of problems solvable by polytime algorithms?
$\rightsquigarrow$ back to *decision* problems.

Some simplifications:

- Only 3 sensible output values: $0, 1, ?$.
- To allow full power of randomization, always allow $Random_A(c) = time_A(c)$, i. e., every step may use a random bit.

Definition 8.1 (ZPP)

ZPP (zero-error probabilistic polytime) is the class of all languages L with a polytime *Las Vegas* algorithm A, i. e., $\Pr\big[A(x) = [x \in L]\big] \geq \frac{1}{2}$ (and $A(x) \neq [x \in L]$ implies $A(x) = ?$), and $time_A(n) = O(n^c)$ as $n \to \infty$ for some fixed c. ◀

Definition 8.2 (BPP and PP)

BPP (bounded-error probabilistic polytime) and PP (probabilistic polytime) is the class of languages with a polytime *bounded-error resp. unbounded-error Monte Carlo* algorithm. ◀

Error Bounds Matter

Remark 8.3 (Success Probability)

From the point of view of complexities, the success probability bounds are flexible:

- BPP only requires success probability $\frac{1}{2} + \varepsilon$, but using *Majority Voting*, we can also obtain any fixed success probability $\delta \in (\frac{1}{2}, 1)$, so we could also define BPP to require, say, $\Pr\big[A(x) = [x \in L]\big] \geq \frac{2}{3}$.
- Similarly for ZPP, we can use probability amplification on Las Vegas algorithms to obtain any success probability $\delta \in (\frac{1}{2}, 1)$.

◀

But recall: this is *not* true for unbounded errors and class PP.
In fact, we have the following result.

Theorem 8.4 (PP can simulate nondeterminism)

$\mathsf{NP} \cup \mathsf{co\text{-}NP} \subseteq \mathsf{PP}$. ◀

$\rightsquigarrow$ Useful algorithms must avoid unbounded errors.

One-sided errors

In many cases, errors of MC algorithm are only *one-sided*.

Example: (simplistic) randomized algorithm for SAT
Guess assignment, output $[\phi \text{ satisfied}]$.
(NB: This is not a MC algorithm, since we cannot give a fixed error bound!)

Observation: No false positives; unsatisfiable ϕ always yield 0.
. . . does this help?

Definition 8.5 (One-sided error Monte Carlo algorithms)

A randomized algorithm A for language L (i. e., for $f(x) = [x \in L]$) is a one-sided-error Monte-Carlo (OSE-MC) algorithm if we have

1. $\mathbb{P}[A(x) = 1] \geq \frac{1}{2}$ for all $x \in L$, and
2. $\mathbb{P}[A(x) = 0] = 1$ for all $x \notin L$.

Definition 8.6 (RP, co-RP)

The classes RP and co-RP are the sets of all languages L with a polytime OSE-MC algorithm for L resp. $\overline{L}$. ◀

Theorem 8.7 (Complementation feasible → errors avoidable)

$\mathsf{RP} \cap \text{co-}\mathsf{RP} \;=\; \mathsf{ZPP}$. ◀

Note the similarly to the open problem $\mathsf{NP} \cap \text{co-}\mathsf{NP} \stackrel{?}{=} \mathsf{P}$;
... a first hint that randomization might not help too much?

8.2 Derandomization

Derandomization

Trivial observation: If $Random_A(n) \le c \operatorname{ld} n$, there are only $2^{Random_A(n)} = n^c$ different computations.

$\rightsquigarrow$ We can simply execute all of them sequentially in polytime!

We can extend this to more random' bits using *pseudorandom generators*, i. e., algorithms that use a limited amount of real randomness and compute from this a much longer sequence of bits that look random (pseudorandom) to *every* efficient algorithm.

It is not proven that such a method exists, but under widely believed assumptions on circuit complexity lower bounds, there is such a pseudorandom generator that allows to derandomize BPP (!)

$\rightsquigarrow$ Current belief is BPP = P . . . and hence BPP = RP = co-RP = ZPP = P (!)

For solving hard problems in theory, randomization does not help at all!

(or: no sufficiently strong lower bound techniques known!)